Online Quadrotor Trajectory Generation and Autonomous Navigation on Point Cloud

Fei Gao and Shaojie Shen

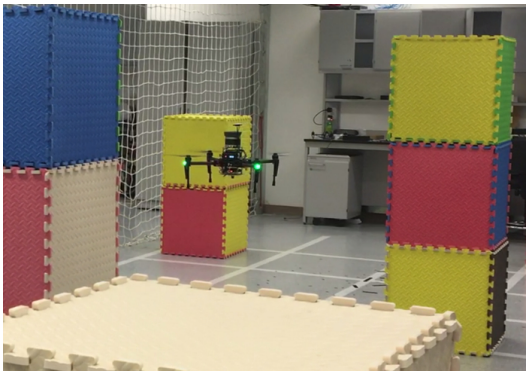
***Abstract*— In this paper, we present a framework for online generation of safe trajectories directly on point cloud for autonomous quadrotor flight. Considering a quadrotor operateing in unknown environments, we use a 3-D laser range finder for state estimation and simultaneously build a point cloud map of the environment. Based on the incrementally built point cloud map, we utilize the property of the fast nearest neighbor search in KD-tree and adopt the sampling-based path finding method to generate a flight corridor with safety guarantee in 3-D space. A trajectory generation method formulated in quadratically constrained quadratic programming (QCQP) is then used to generate trajectories that constrained entirely within the corridor. Our method runs onboard within 100 milliseconds, making it suitable for online re-planning. We integrate the proposed planning method with laser-based state estimation and mapping modules, and demonstrate the autonomous quadrotor flight in unknown indoor and outdoor environments.**

I. Introduction

Micro aerial vehicles (MAVs), especially quadrotors, have drawn increasing attention for security and safety missions thanks to their superior mobility in complex environments that are inaccessible or dangerous for human or other ground vehicles. In search and rescue missions, quadrotors should be able to online generate and execute smooth and safe trajectories from a start position to a target position, while avoiding unexpected obstacles. In this paper, we build on top of the state-of-the-art laser-based solution [1] for state estimation and mapping in large-scale unknown environments, and propose an online trajectory generation method that directly operates on point cloud for safe aerial navigation. We integrate all modules into a complete system and present online experimental results.

Given estimation of the vehicle state, many approaches have been proposed to convert range measurements generated by onboard sensors into a global map. Representative methods include voxel grid [2], octomap [3], elevation map [4], etc. Each of these methods has pros and cons in particular environments. For instances, voxel grids are good for fine-grain representation of small volumes, but the storage complexity scales badly. Octomaps are memory efficient when representing environments with large open spaces, at the expense of costly maintenance of the map structure. Elevation maps is good for representing man-made structures consist of mostly vertical walls, but less efficient when representing natural scenes. In this paper, instead of using post-processed maps, we opt to plan trajectories directly on globally-registered point clouds. This bypasses the costly map building process and achieves the best adaptivity for different environments, which is one of the key requirements for search and rescue missions.

All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. fgaoaa@connect.ust.hk, eeshaojie@ust.hk



(a) Autonomous flight in unknown indoor environment

(b) Autonomous flight in unknown outdoor environment

Fig. 1. Our quadrotor testbed equipped with a Velodyne VLP-16 3-D LiDAR, an IMU and a sonar. Onboard computation is an Intel NUC with i5-4250U CPU running at 1.30 GHz. We demonstrate autonomous flight through complex indoor and outdoor environments. Video is available at http://www.ece.ust.hk/~eeshaojie/ssrr2016fei.mp4

As presented by Mellinger et al. [5], a quadrotor system enjoys the differential flatness property, making it possible to reduce the full state space to the 3-D position and yaw angle and their derivatives. Smooth piecewise polynomial trajectories of 3-D position with bounded derivatives therefore can be followed by a properly designed geometric controller. We utilize this property and design a novel method for generating collision-free smooth trajectories directly on point cloud using convex optimization. Our method employs a randomized flight corridor generation method utilizing fast neighbor search on KD-tree, followed by an optimization-

based polynomial trajectory generator. We summarize our contributions as follows:

1) A sampling-based, rapidly exploring random graph (RRG) method, combined with an A* path finding algorithm, for finding a collision-free flight corridor that consists of multiple overlapping free space balls.
2) A quadratically constrained quadratic programming (QCQP)-based trajectory generation method that operates within the flight corridor with safety guarantee.
3) Real-time implementation with online map update and trajectory (re)generation within 100 milliseconds, enabling online navigation and exploration of unknown complex environments.
4) Integration with state estimation and mapping modules and presenting online experiments in large-scale indoor and outdoor environments

We discuss relevant literature in Sect. II, and introduce the overall system architecture in Sect. III. Laser-based state estimation and mapping, which build the prerequisite for our method, are presented in Sect. IV. Our corridor generation and trajectory optimization methods are detailed in Sect. V and Sect. VI respectively. In Sect. VII, implementation details, as well as simulation and experimental results are presented. The paper is concluded in Sect. VIII.

II. Related Work

There are a large number of algorithms and applications on robotic motion/trajectory planning, ranging from sampling-based methods [6], to optimization-based methods [5]. Here we provide an overview of approaches that are relevant to the task of quadrotor trajectory planning.

The approach in [7] combines a fixed final state free final time controller with the rapidly-exploring random tree* (RRT*) method to ensure asymptotic optimality. In this way, closed-form solutions for optimal trajectories can be derived. Karaman and Frazzoli [8] proposed the RRG method, which is an extension of the RRT algorithm, as it connects new samples not only to the nearest node but also all other nodes within a ball. Another method combining the RRG and the belief roadmap was proposed by Bry and Roy et al. [9], in which a partial ordering is introduced to trade-off the belief and the distance.

Control-based approaches can also be used to solve planning and obstacle avoidance problems. [10] considered a linear quadrotor model, and proposed the linear quadratic gaussian (LQG) obstacle set to represent target states that lead to collision. Safe control commands can be generated as long as target states are outside of this set. A receding horizon control policy was proposed in [11] for similar purpose.

Mellinger et al. [5] pioneered a minimum snap trajectory generation algorithm. There is a large body of follow-up works. A mixed integer quadratic programing (MIQP) is proposed in [12] to enforce avoidance of both static obstacles and other agents in a heterogeneous quadrotor team. Unconstrained quadratic programming (QP) with a closed-form solution is formulated in [13], in which RRT* is used to find a feasible straight-line path, followed by a waypoints-based trajectory. If the trajectory intersects with obstacles, additional intermediate waypoints are added and the trajectory is generated again.

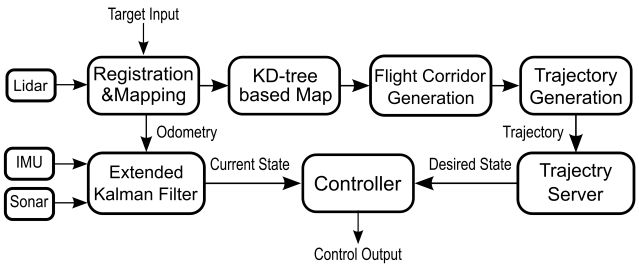


Fig. 2. The architecture of our quadrotor system. The point cloud registration generates odometry at 20 Hz using the method proposed in [1]. Registered point cloud stored in a KD-Tree data structure is updated at 10 Hz. Measurements from a 100 Hz onboard IMU is fused with the laser odometry and height measurements from a sonar using an EKF. The flight corridor generation and trajectory generation runs on-demand and can usually be completed within 100 milliseconds.

Trajectory generation assisted by geometric constraints has been becoming popular recently. In [14], iterative regional inflation by semi-definite programming (IRIS), a recently developed convex segmentation method, is used to approximate the free space with convex regions. Then a sums-of-squares (SOS) programming is used to assign the smooth trajectories to these convex regions. Chen et al. [15, 16] proposed an online method that utilizes efficient operations in the octomap data structure for online generation of a flight corridor. Quadratic programming is used for the generation of safe and dynamically feasible trajectories.

Our approach belongs to the same class of geometrically-constrained trajectory optimization methods, except that we propose to bypass the mapping process and directly operate on point cloud. In [17], a path planning algorithm that takes point cloud as the input is proposed. This method utilizes tensor voting to reconstruct surfaces and find local flat regions for ground vehicle navigation. However, this algorithm is fundamentally different from ours as we find trajectories in full 3-D space.

III. Overview

We begin by presenting the overall architecture of our autonomous quadrotor system (Fig. 2). Range measurements from the Velodyne VLP-16[1] 3-D LiDAR are used for pose estimation [1] and generation of the globally-registered point cloud map. During this process, a generalized ICP-based method is used for frame-to-frame 6-DOF motion estimation. This incremental motion is used as the initial guess for a second stage of frame-to-map alignment to achieve global consistency, also using an ICP-based algorithm. KD-Tree is used as the data structure for map storage and for nearest neighbor search. To obtain the high-rate state estimates for feedback control, the laser-based pose estimation is

[1] http://velodynelidar.com/vlp-16.html

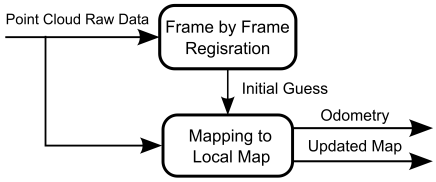


Fig. 3. Pipeline of laser-based perception modules. State estimation runs at 20 Hz, while the point cloud map is updated at 10 Hz. We only keep a fixed size local map in order to bound the computation and storage complexity.

fused with IMU and sonar measurements using an extended Kalman filter (EKF).

Using the point cloud, a sampling-based method is adopted to generate a collision-free flight corridor. In this method, we sample and connect free space with shape of balls which has locally maximum radius. Radius search can be done very efficiently using the KD-tree structure. The corridor is thus a series of connected balls. Using the flight corridor as geometric constraints, an optimization-based method is used to generate collision-free piecewise polynomial trajectories that fit entirely within the corridor. The corridor and trajectory generation can be done within 100 milliseconds, making it possible for online trajectory (re)generation for avoidance of unexpected obstacles. Finally, a feedback controller is used for trajectory tracking.

IV. Laser-based State Estimation and Mapping

We now present our implementation of a laser-based state estimation and point cloud mapping method. Our approach is adapted from [1] and it forms the perception foundation for a complete navigation system that is able to utilize our trajectory generation approach.

The pipeline of our laser-based state estimator is shown in Fig. 3. We extract edges and surfaces from each 360 degree 3-D range measurements. A generalized iterative closest point (ICP) scan matching algorithm is performed to do the pose estimation between scan S_i and S_j, with steps summarized as follows:

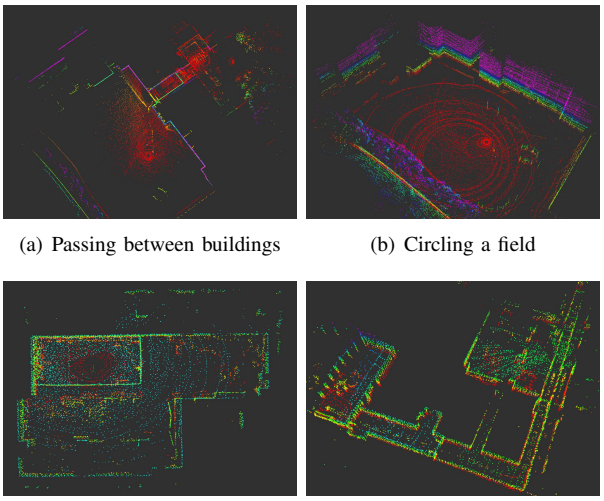
- For each 3-D point on an edge of S_i, we search for the closest two points in S_j. The distance (d_s) from a point ($\mathbf{x}_i^0$) to a line (formed by $\mathbf{x}_j^1$ and $\mathbf{x}_j^2$) is calculated as

$$d_e = \frac{\left|(\mathbf{x}_j^2 - x_j^1) \times (\mathbf{x}_j^1 - \mathbf{x}_i^0)\right|}{\left|\mathbf{x}_j^2 - \mathbf{x}_j^1\right|}, \tag{1}$$

- For each 3-D point on a surface of S_i, we search for the closest three points in S_j that are not co-linear, and calculate the distance (d_s) from a point ($\mathbf{x}_i^0$) to a surface (formed by $\mathbf{x}_j^1$, $\mathbf{x}_j^2$, $\mathbf{x}_j^3$) as:

$$d_s = \frac{(\mathbf{x}_j^2 - \mathbf{x}_j^1) \times (\mathbf{x}_j^3 - \mathbf{x}_j^1)}{\left|(\mathbf{x}_j^2 - \mathbf{x}_j^1) \times (\mathbf{x}_j^3 - \mathbf{x}_j^1)\right|} \cdot (\mathbf{x}_i^0 - \mathbf{x}_j^1) \tag{2}$$

- We stack the distance from each feature point, and use the Levenberg-Marquardt method [18] to minimize the sum of distances for both lines and surfaces to obtain the 6-DOF relative pose.



(a) Passing between buildings (b) Circling a field

(c) Mapping the laboratory (d) Moving through the lobby

Fig. 4. Snapshots of point cloud maps built online.

We use the same ICP algorithm for both scan-to-scan ($i = j - 1$) and scan-to-map (S_j is the global point cloud map) alignment to obtain the 6-DOF laser pose. The result of scan-to-scan alignment is used as the initial guess for the scan-to-map alignment, after which points from the current scan is added into the map $\mathcal{M}$. Additionally, we highlight two features of our point cloud map:

1) We maintain constant computation complexity by limiting the total number of points. We bound the volume that the point cloud can spread, and restrict the spatial density of points. This results in a local map that slides with the UAV.

2) We dynamically update the weight (w) for each point in the map in order to clear out moving objects and outlier measurements. At the time of the i^{th} scan, we can define the weight update function for the point $\mathbf{x}_i^k$ as follows:

$$w(x_{i+1}^k) = \begin{cases} \min(w(\mathbf{x}_i^k) - \alpha + \beta, \lambda_H), & \text{if } \exists\, \mathbf{x}_{i+1}^k, s.t. \\ & \|\mathbf{x}_i^k - \mathbf{x}_{i+1}^k\| \leq \delta \\ \max(w(\mathbf{x}_i^k) - \alpha, \lambda_L), & \text{otherwise} \end{cases} \tag{3}$$

where α and β are parameters that control the rate of decreasing and increasing of the weight. δ is a small number for determining whether the same point has been observed. λ_H and λ_L are upper and lower bounds of the weight, at which the point is considered as permanently added or removed from the map. Points that are observed repeatedly will eventually be fixed in the map, while points that received few observations, such as points on moving objects or outliers, will be removed after a number of updates. Snapshots of point cloud maps are shown in Fig. 4.

To achieve the high-rate state estimation required for feedback control. We use an EKF to fuse laser pose estimation with IMU and sonar measurements. For compensating the delay in each sensor, we adopt a ring-buffer method presented by Lynen [19]. In our filter, acceleration and angular velocity measurement from the IMU are used as the

process model, while odometry from LiDAR and sonar are both used as relative measurement models for 6-DOF pose and height. Our 3-D LiDAR has only 30 degrees of vertical coverage, making it prone to failure in altitude estimation in environments containing only vertical structures. The addition of a downward facing sonar solves this issue.

V. Flight Corridor Generation on Point Cloud

A. KD-Tree-Based Environment Representation

Our laser-based state estimation algorithm is performed utilizing the KD-tree [20] data structure for fast nearest neighbor points search. A balanced KD-tree with N points can be constructed in $O(NlogN)$ time with a space complexity of $O(kN)$, and the M nearest neighbors search has a time complexity of $O(MlogN)$. For state estimation, the KD-tree is useful for point cloud registration. But from a point of view of planning, the nearest neighbor search can also be used to find the distance between an arbitrary unoccupied location in the 3-D space and its nearest point (obstacle) in the map. This distance indicates the safe radius with respect to that location, from which a safe region in the shape of a ball can be generated. Note that the KD-tree is a static data structure, which means inserting points into a KD-tree dynamically will make it unbalanced. Therefore, every time the map is updated, we reconstruct the KD-tree. However, as shown in Fig. 13, the time required for reconstructing a KD-tree with 80,000 points is acceptable for real-time applications.

B. Generation of A Graph of Ball-Shape Safe Regions

As presented in Sect. V-A, the safe radius of an arbitrary point in the map can be found by fast nearest neighbor search in the KD-tree. Based on this property, we adopt a random sampling-based method to generate corridors connecting the starting point and the target point. To begin, we use RRG to construct a randomly sampled graph with vertices being the centers of ball-shape safe regions, and edges being the connectivities between safe regions. The flight corridor then consists of a sequence of connected ball-shape safe regions, where the location, size of the regions, as well as the connectivities between them are determined by this RRG.

As shown in Algorithm. 1, a graph $\mathcal{G}$ is initialized with only the starting node n_s. A node n in the graph represents a free space ball with two properties, the 3-D location $n.\mathbf{c}$ and the radius $n.r$. When a new location $\mathbf{c}_r$ is randomly sampled, **nearest**$(\mathbf{c}, \mathcal{G})$ finds its nearest node n_c in the graph. Our algorithm ensures the generation of a graph with edges being collision-free connection between ball-shape safety regions. The **intersect**() function generates a ray from $\mathbf{c}_r$ to $n_c.\mathbf{c}$, and returns the intersection point of the ray with n_c's sphere. We set this point as the center of a new node $n_n.\mathbf{c}$. We perform radius search **radius_search**$(n_n.\mathbf{c}, \mathcal{M})$ to find the largest ball-shape safe region centered at $n_c.\mathbf{c}$ against the point cloud map $\mathcal{M}$. If $n_n.r$ is sufficiently large, the node will be added to the graph and connected with n_c. We also find all nodes at the neighbor of n_n using a nearest neighbor within $\mathcal{G}$. We connect them with n_n if the overlapping volume, which can be calculated very conveniently in closed form, is larger than a safety margin to allow the quadrotor to safely travel through two balls.

Algorithm 1 RRG on Point Cloud

```
Requires M
G ← n_s, i ← 0
while i ≤ N do
  c_r ← sample()
  n_c ← nearest(c_r, G)
  n_n.c ← intersect(c_r, n_c.c)
  n_n.r ← radius_search(n_n.c, M)
  if n_n.r > λ_r then
    G ← G ⋃ n_n
    connect(n_c, n_n)
    for all n_j ∈ neighbor(n_n, G) do
      if overlap_volume(n_j, n_n) > λ_v then
        connect(n_j, n_n)
      end if
    end for
  end if
  i ← i + 1
end while
```

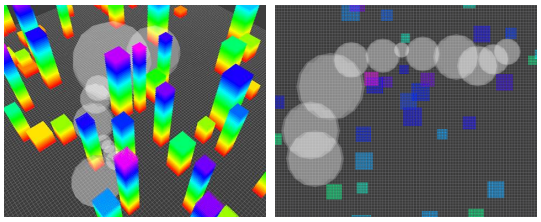
Since every time we sample a new point, we create a new node only centered on the nearest node's sphere, the expansion rate of the graph varies according the sparsity of obstacles, making it particularly efficient for environments with large volumes of free spaces. The RRG stops based on a predefined iteration limit N.

C. Generation of the Flight Corridor

Having the connected graph of safe regions, we can utilize search-based algorithms such as A* search method to find a path from the starting position to the target position on the graph. Note that since the start position and target position might be contained in one of many overlapping safe regions, there might be multiple choices of start and target regions. We opt to use a heuristic to pick the target region as the one nearest to the starting position in Euclidean metric. The flight corridors found by A* are illustrated in in Fig. 5. Note that the algorithm we use here has the characteristic of asymptotic optimality. As the number of safe region samples increases, it will eventually densely fills the total free space, making it equivalent to running search-based algorithm on uniformly and finely discretized voxel grids.

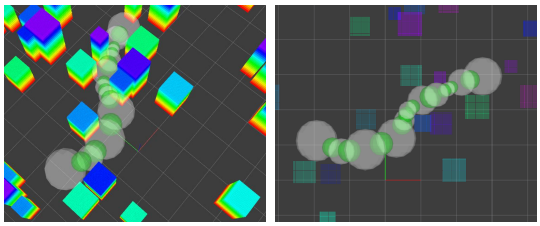
VI. Optimization-Based Trajectory Generation with Safety Guarantee

In this section, we present our optimization-based method for generating smooth trajectories that are constrained by the safe flight corridor (Sect. V). Thanks to the differential flatness property of the quadrotor model, the full state space of the quadrotor system $\{x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r\}$ is reduced to the 3-D position and the yaw angle$\{x, y, z, \psi\}$ and their derivatives in which the trajectory is generated. A



(a) Flight corridor through obstacles (b) Downward-facing view of the same flight corridor

Fig. 5. Flight corridor found by our ball-shape RRG and A* algorithm. The overlapping and connected white balls are the ball-shape safe regions that are used to form the flight corridor. Color blocks are random 3-D obstacles.



(a) Free space inflation in connected regions (b) Downward-facing view of the same flight corridor

Fig. 6. Flight corridor with connected region inflation. Markers in this figure can be interpreted similarly as in Fig. 5. Additionally, green spheres represent the inflated connection regions in the flight corridor.

geometric tracking controller [21] can be applied for feedback control. Since we have a 360-degree LiDAR onboard our quadrotor, we skip the planning of the yaw angle.

A. Polynomial Trajectory Generation

The trajectory consisting of piecewise polynomials are parametrized to the time variable t in each dimension μ out of *x*, *y*, *z*. The M-segment trajectory of one dimension can be written as follows:

$$f_\mu(t) = \begin{cases} \sum_{j=0}^{N} p_{1j}(t-T_0)^j & T_0 \le t \le T_1 \\ \sum_{j=0}^{N} p_{2j}(t-T_1)^j & T_1 \le t \le T_2 \\ \quad\vdots & \quad\vdots \\ \sum_{j=0}^{N} p_{Mj}(t-T_{M-1})^j & T_{M-1} \le t \le T_M, \end{cases} \tag{4}$$

where p_{ij} is the j^{th} order polynomial coefficient of the i^{th} segment of the trajectory. $T_1, T_2, ..., T_M$ are the end time of each segment, with total time of $T = T_M - T_0$. The polynomial coefficients are computed by minimizing a cost function of the k_ϕ^{th} derivative along the trajectory.

The cost function is the integral of the square of the k_ϕ^{th} derivative. Instead of formulating the cost function for each dimension as in [5], in this paper, the coefficients in all *x*, *y*, *z* dimensions are coupled into one single equation:

$$J = \sum_{\mu\in\{x,y,z\}} \int_0^T \left(\frac{d^{k_\phi} f_\mu(t)}{dt^{k_\phi}} \right)^2 dt, \tag{5}$$

The objective function can be written in a quadratic formulation $\mathbf{p}^T\mathbf{Q}_o\mathbf{p}$, where $\mathbf{p}$ is a vector containing all polynomial coefficients (a_{ij}) in all three dimensions of *x*, *y*, *z*, and $\mathbf{Q}_o$ is the Hessian matrix of the objective function. For briefly, we omit the detailed construction of the cost function. We must also enforce a number of constraints to ensure the smoothness and safety of the trajectory:

1) Waypoint Constraints: If needed, we can fix waypoints (such as start and target positions) and their first k_ϕ - 1 derivatives d_{ik}. If a fixed waypoint exists at the time of T_i, we have:

$$f_\mu^{(k)}(T_i) = d_{ik}, \tag{6}$$

2) Continuity Constraints: The trajectory must be continuous at all the k^{th} derivatives at the connecting point between two polynomial segments, where $0 \le k \le k_\phi - 1$:

$$\lim_{t\to T_i^-} f_\mu^{(k)}(t) = \lim_{t\to T_i^+} f_\mu^{(k)}(t), \tag{7}$$

3) Intersection Between Safe Regions: At the end time of each polynomial segment, the position of the trajectory must be inside the connected ball-shape safety regions:

$$\begin{cases} \sum_{\mu\in\{x,y,z\}} (f_\mu(T_i) - \mathbf{c}_i^\mu)^2 & \le r_i^2 \\ \sum_{\mu\in\{x,y,z\}} (f_\mu(T_i) - \mathbf{c}_{i+1}^\mu)^2 & \le r_{i+1}^2, \end{cases} \tag{8}$$

where $T_i (i = 1, 2, ..., M)$ are the end time of the piecewise polynomial, $\mathbf{c}_i$ is the ball center and r_i is the radius.

The first and second constraints can be reformulated as linear equality constraints ($\mathbf{A}_{eq}\mathbf{p} = \mathbf{b}_{eq}$), while the third constraint must be written in a quadratic inequality constraints form ($\mathbf{p}^T\mathbf{Q}_i\mathbf{p} + \mathbf{a}_i\mathbf{p} \le \mathbf{b}_i$). Thus, the trajectory generation problem can be reformulated as a quadratically constrained quadratic programming (QCQP) problem:

$$\begin{aligned} \min \quad & \mathbf{p}^T\mathbf{Q}_o\mathbf{p} \\ \text{s.t.} \quad & \mathbf{p}^T\mathbf{Q}_i\mathbf{p} + \mathbf{a}_i\mathbf{p} \le \mathbf{b}_i, \;\; i = 0, \cdots, M-1 \\ & \mathbf{A}_{eq}\mathbf{p} = \mathbf{b}_{eq}, \end{aligned} \tag{9}$$

In practice, small numerical errors in high order polynomial coefficients may cause serious errors in the overall trajectory. We therefore adopt the method proposed by Richter et al. [22] in which the derivatives at each end time of trajectory segments are calculated, then mapped to the original polynomial coefficients. Additionally, we use the time normalization method proposed by Mellinger et al. [5] to avoid large numerical errors in the Hessian matrix caused by the large time value in each segment.

B. Effects on Adjustment of Segment Connecting Points

As is shown by Chen et al. [15], a flight corridor with large overlapping regions can introduce a large space for the transition between polynomial segments. This provide an opportunity to implicitly adjust the time allocation. Larger overlapping regions are therefore desired as they provide more freedom for the optimization to reduce the trajectory cost. In this paper, we inflate the connected region between

two consecutive balls into a new ball. As shown in Fig. 6, the green spheres are the inflated connection regions with diameters being the chord of two intersecting balls. Thus we can modify our quadratic constraint as

$$\sum_{\mu\in\{x,y,z\}} (f_\mu(T_i) - \mathbf{c}_j^\mu)^2 \leq r_j^2, \tag{10}$$

where $\mathbf{c}_j$ and r_j are center and radius of the inflated ball. This modification not only inflates more space for the adjustment of segment connecting points, but also halves the number of quadratic constraints.

C. Enforcing Safety and Dynamical Constraints

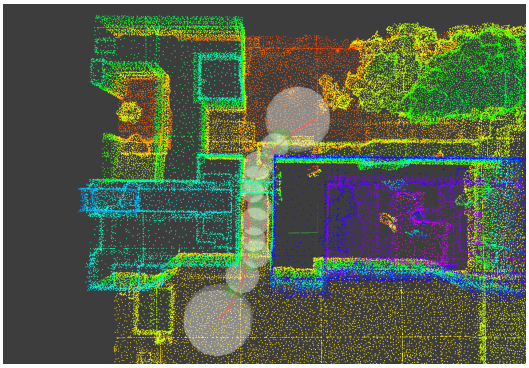
The trajectory must be fully constrained within the flight corridor to ensure collision avoidance. We also need to impose maximum velocity and acceleration constraints to ensure dynamical feasibility. In the previous section we enforced the trajectory remaining within the corridor at the end time of each segment, and now we should make sure it will not go beyond the corridor at any other time. We adopt Chen's method [15], which proves that a finite number of point constraints that are iteratively added at the polynomial's extrema lead to fully bounded trajectories. The drawback of applying this method in our algorithm is that we might constrain the extrema into the original overlapping safe regions rather than the inflated connection region. By doing so, the effect on segment connecting point adjustment may be reduced, resulting in less optimal, but still safe and feasible trajectories.

VII. Results

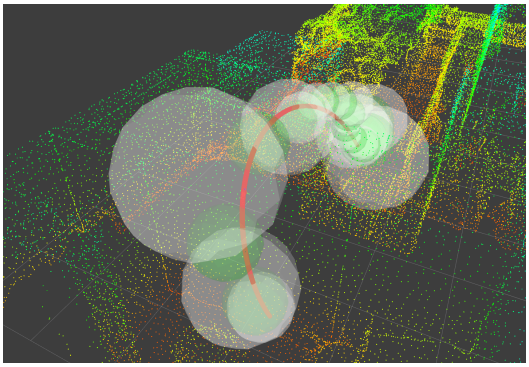
Simulation results and experiments of indoor and outdoor autonomous flight are presented in this section. Our trajectory generation module is implemented in C++11 using a general convex solver Mosek[2] under a free academic license and the sparse matrix library in Eigen. The simulation is done on a laptop equipped with a 2.20 GHz Intel Core i5-5200U CPU and 8 GB RAM. The flight experiments are done on a DJI M100 quadrotor (Fig. 1) equipped with an Intel NUC (dual-core CPU i5-4250U at 1.30 GHz and 16GB RAM). The simulation is done with a known map of our university built by an offline mapping software. Autonomous flight experiments are done in unknown environments using only online sensor measurements. The point cloud maps built online is visualized in Figs. 10 and 11.

A. Simulation Results

The 3-D point cloud map used in the simulation is built by Altizure[3] using the data collected from high altitude by a drone. The map contains more than 400,000 points, covering 160 meters in horizontal directions and 40 meters in the vertical direction. We limit the altitude of the simulated quadrotor to 20 meters to mimic the practical scenario in close-proximity search and rescue missions. In Fig. 7(a), the collision-free trajectory with a total length of 95.138 meters is computed in 74.3 milliseconds on our standard laptop. Fig. 7(b) shows another scenario, where the length of the trajectory is 60.884 meters, and is computed in 60.8 milliseconds. The simulation shows that our algorithm is sufficiently fast in large-scale environments and the computation complexity scales favorable with respect to travel distance.



(a) The simulated quadrotor traveling between buildings



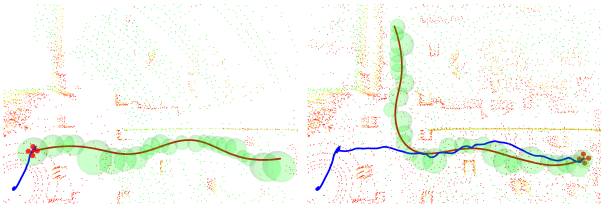
(b) The simulated quadrotor flies above the wall

Fig. 7. Simulated flight in a known map (Sect. VII-A). The map is built offline and contains more than 400,000 points. Red curves are the generated smooth trajectory. White spheres and green spheres are the flight corridor (Sect. V) and the inflated connection regions (Sect. VI-B), respectively.

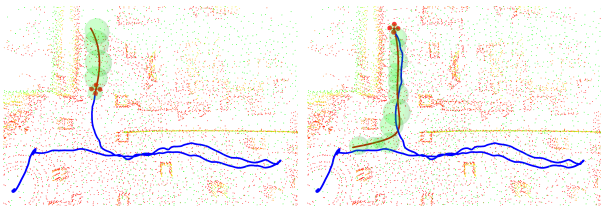
B. Autonomous Indoor Flight

The indoor experiment is done in our laboratory with randomly placed obstacles (Fig. 1(a)). The quadrotor is commanded to fly through a number of predefined waypoints. State estimation, mapping, and trajectory generation are all performed online without any prior information about the environments. The Velodyne 3-D LiDAR that runs at 20 Hz outputs 15,000 points in each scan. State estimation runs at the same rate as the sensor, while the point cloud is updated at 10 Hz. After map update, the quadrotor checks the collision status of the current trajectory and re-generates if necessary. We set a safety margin considering the size of the quadrotor and reserve 0.15 meters in the radius for each safe region in the flight corridor to account for control error. In the experiment, three targets are sent to the quadrotor sequentially, and four trajectory (re)generation occurred due to the online detection of obstacles (Figs. 8 and 9).

[2] https://www.mosek.com
[3] https://www.altizure.com/explore



(a) 1^{st} target, 1^{st} trajectory (b) 2^{nd} target, 2^{nd} trajectory



(c) 2^{nd} target, 3^{rd} trajectory (d) 3^{rd} target, 4^{th} trajectory

Fig. 8. Autonomous indoor flight experiment (Sect. VII-B). The generated desired trajectories are in red, while actual trajectories that the quadrotor have already flown are in blue. Safe flight corridor is shown as green balls.

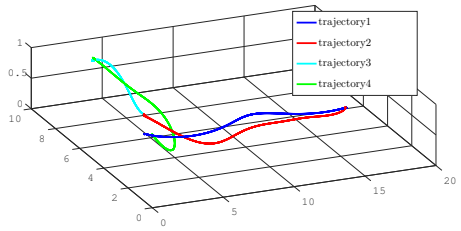


Fig. 9. Trajectories generated during indoor autonomous flight (Sect. VII-B). different trajectories are indicated in different colors. Trajectory re-generation occurred when flying towards the second target due to the detection of new obstacles.

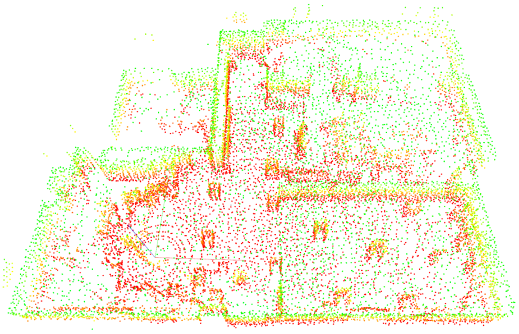
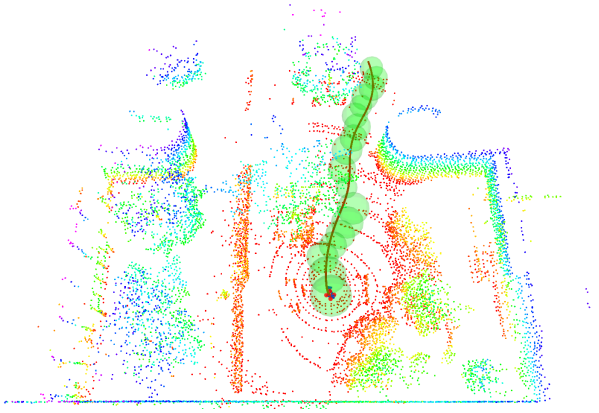


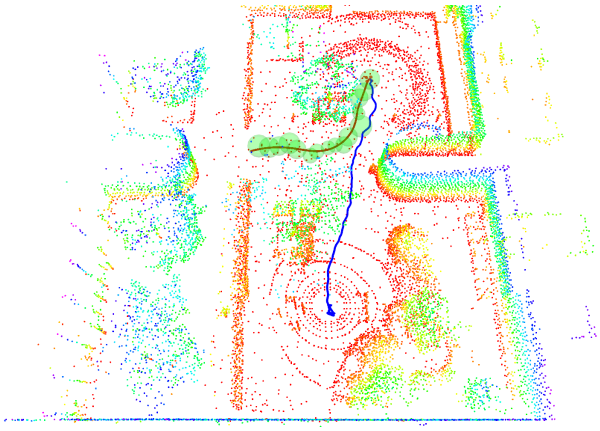
Fig. 10. Final point cloud map built from autonomous indoor flight (Sect. VII-B).

C. Autonomous Outdoor Flight

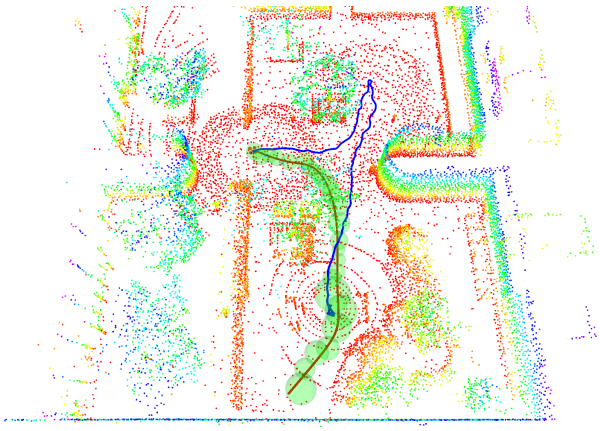
The outdoor experiment is done in a complex environment with obstacles of various shapes and sizes, such as trees, bushes and tables. A snapshot of the flight is shown in Fig. 1(b). Three trajectory generation occurred. Results are shown in Figs. 11 and 12. For each trajectory generation in both indoor and outdoor flights, the time consumed for KD-tree construction, RRG sampling, A* search and trajectory generation are shown in Fig. 13. In our method, although the KD-tree is reconstructed after every map update, the time consumption is still acceptable. Additionally, the time spent in sampling is proportional to the number of samples, making it a trade-off between the optimality of the corridor and the computation complexity.



(a) 1^{st} target, 1^{st} trajectory



(b) 2^{nd} target, 2^{nd} trajectory



(c) 3^{rd} target, 3^{rd} trajectory

Fig. 11. Visualization of autonomous outdoor flight (Sect. VII-C). Markers can be interpreted in the same way as Fig. 8

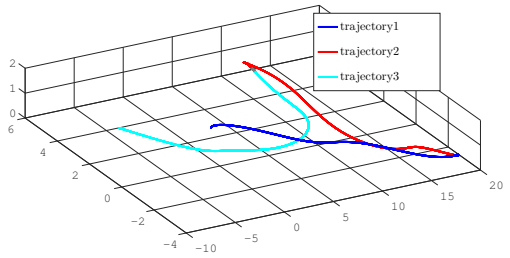


Fig. 12. Trajectories generated in the autonomous outdoor flight (Sect. VII-C). Markers in this figure have the same meaning as in Fig. 9

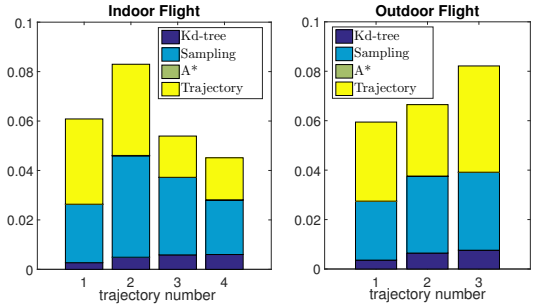


Fig. 13. Time consumption for each system module for both indoor and outdoor flights (Sect. VII-B and Sect. VII-C). Note that the computing time in A* is too short to be noticeable in the figure.

VIII. Conclusion and Future Work

We propose a method for online trajectory generation in unknown environments for a quadrotor. Our method utilizes the point cloud map to directly find a collision-free flight corridor, followed by an optimization-based trajectory generation method for smoothness and safety guarantee. Our method is implemented onboard a quadrotor equipped with a LiDAR, and is suitable for fast online re-planning for avoidance of unexpected obstacles.

In the future, we aim to improve our work to achieve large-scale planning and exploration, which targets at autonomously explore unknown large-scale environments without any pre-specified targets. Additionally, although the KD-tree is capable to perform fast nearest neighbor search, it has an obvious drawback that every time we update our map, we have to reconstruct the KD-tree. We plan to utilize more advanced data structures for incremental update of the point cloud map and the randomized flight corridor search graph.

References

[1] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Proc. of Robot.: Sci. and Syst.*, UCB, USA, July 2014, pp. 109–111.

[2] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[3] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Anchorage, AK, US, May 2010.

[4] S. Choi, J. Park, E. Lim, and W. Yu, "Global path planning on uneven elevation maps," in *Proc. of the IEEE Intl. Conf. on Ubiquitous Robot. and Ambient Intelligence*, Daejeon, Korea, Nov. 2012.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 2520–2525.

[6] J. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. of the IEEE Control and Decision Conf.*, Orlando, FL, Dec. 2011, pp. 3276–3282.

[7] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013, pp. 5054–5061.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[9] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 723–730.

[10] J. van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "LQG-Obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Saint Paul, MN, May 2012, pp. 346–353.

[11] M. Watterson and V. Kumar, "Safe receding horizon control for aggressive mav flight with limited range sensing," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Hamburg, Germany, sept. 2015, pp. 3235–3240.

[12] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Saint Paul, MN, May 2012, pp. 477–483.

[13] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Intl. Sym. of Robot. Research*, Singapore, Dec. 2013.

[14] D. Robin and T. Russ, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* Seattle, Washington, USA: IEEE, May 2015, pp. 42–49.

[15] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016, URL http://www.ece.ust.hk/~eeshaojie/icra2016jing.pdf.

[16] J. Chen, K. Su, and S. Shen, "Real-time safe trajectory generation for quadrotor flight in cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Biom.*, Zhuhai, China, Aug. 2015, URL http://www.ece.ust.hk/~eeshaojie/robio2015jing.pdf.

[17] M. Liu, "Robotic online path planning on point cloud," *IEEE Transactions on Cybernetics*, 2015, in press.

[18] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, Nov. 2013, pp. 3923–3929.

[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[21] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Proc. of the IEEE Control and Decision Conf.*, Atlanta, GA, Dec. 2010, pp. 5420–5425.

[22] R. Charles, B. Adam, and R. Nicholas, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Intl. Sym. of Robot. Research*, Singapore, Dec. 2013.